A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints

David Bredström[a] and Mikael Rönnqvist[b]

[a]Linköping University, Linköping, Sweden
[b]Norwegian School of Economics and Business Administration, Bergen, Norway

Abstract

In this paper we present a branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints. The synchronization constraints are used to model situations when two or more customers need simultaneous service. The synchronization constraints impose a temporal dependency between vehicles, and it follows that a classical decomposition of the vehicle routing and scheduling problem is not directly applicable. With our algorithm, we have solved 44 problems to optimality from the 60 problems used for numerical experiments. The algorithm performs time window branching, and the number of subproblem calls is kept low by adjustment of the columns service times.

Keywords: Routing, Scheduling, Synchronization, Branch and Price

1 Introduction

Combined vehicle routing and scheduling with time windows arises in many applications and there is an extensive and wide research literature on Operations Research (OR) models and methods Temporal constraints within a route for one vehicle, in addition to time windows, frequently occurs in well

known problems such as the dial-a-ride and the pickup and delivery problems. However, the problem with vehicle dependencies has been given much less attention in the literature despite its wide range of practical applications. A typical application is when two vehicles must meet at a specific point at the same time or when a vehicle cannot pick up a load until another vehicle has delivered that same load. The main goal of this paper is to develop a general branch and price framework for the routing and scheduling with time windows and synchronization constraints based on column generation and time window branching. The synchronization constraints introduced allow for imposing pairwise synchronization between customer visits. Additional temporal constraints in the form of vehicle independent precendece constraints can be modeled and considered in the framework without any major modifications.

Given a fleet of vehicles available in a depot, and a set of customers to be serviced within their respective prescribed time window, the objective for the vehicle routing and scheduling problem (VRSP-TW) is for example to minimize the total traveling time. Both heuristic and exact solution methods have been suggested for solving applications of the VRSP-TW, see e.g. the survey in Desrosiers et al. (1995). The VRSP with a single vehicle and precedence constraints is commonly seen as a traveling salesman problem with precedence constraints. Fagerholt and Christiansen (2000) use the single vehicle VRSP-TW with additional allocation constraints to solve a subproblem arising in a ship scheduling application. If we introduce capacity constraints to the VRSP-TW, depending on the precedence constraints, we get a pickup and delivery problem with time windows (PDP-TW) which is an exhaustively studied problem, see e.g. Desrosiers et al. (1995). Sigurd et al. (2004) use precedence constraints for an application that arises in live animal transport.

In the pickup and delivery and the dial-a-ride problems, the precedence constraints are limited to precedence within a route for a single vehicle. A related problem is the job shop scheduling problem (JSP), where each job is defined by a set of ordered activities and each activity is normally to be executed on one predefined resource. All activities for one job are not bound to one resource and the precedence constraints therefore span over multiple resources, as opposed to the pickup and delivery and the dial-a-ride problems. Beck et al. (2003) study the differences between VRP and JSP and apply both vehicle routing and scheduling techniques to VRPs. In their study, they include vehicle independent precedence constraints to the VRP and observe that the routing techniques they use have difficulty finding feasible solutions, while the scheduling techniques find feasible solutions to

all the studied problem instances.

In the combined vehicle and crew scheduling problem for urban mass transit systems, drivers are allowed to change bus at so called relief points. Commonly, as seen in Haase et al. (2001) and the work of Freling et al. (2003), the arrival time at a relief point is defined by a timetable and therefore the synchronized arrival of bus drivers is implicitly considered. In the homecare scheduling problem presented in Eveborn et al. (2006) there is a required synchronization of staff visits to customers. The model for the periodic routing and airline fleet assignment problem, presented in the paper by Ioachim et al. (1999), has temporal constraints that define the same departure time for pairs of flights, which is a set of synchronization constraints in the sense used in this paper.

Bredström and Rönnqvist (2006) emphasize the importance of the temporal synchronization and precedence constraints found in several real world applications. In this paper we develop a branch and price algorithm influenced by the algorithm introduced for the fleet assignment and routing problem in the paper by Ioachim et al. (1999). Ioachim et al. develop a multi-commodity flow formulation and a solution method based on a side constrained set partitioning formulation, which they solve with column generation in a branch-and-bound framework. The synchronization requirements are for schedules to have the same departure time for certain groups of flights, but on different days over a weekly horizon. The side constraints formulate the requirements using coefficients for departure times. They experience the disadvantage that "a large number of column generation iterations was used for very small variations on the synchronized departure times". To reduce the number of iterations, they introduce a tolerance in the side constraints to allow for limited asynchronous departures. They note that too large tolerance values may result in suboptimality and smaller values may require longer CPU time. The dual prices from the side constraints in their formulation give rise to linear node costs (for the time variables) in the column generator subproblem. To solve the subproblem, they use the exact algorithm presented in Ioachim et al. (1998). The branching is performed firstly on the time windows, and secondly on the flow variables in the subproblem.

In this work, we relax the synchronization constraints from the set partitioning model and satisfy these constraints implicitly in the branch and bound framework. This is done by repeated adjustment of the schedule times (in this case arrival times) and by branching on time windows. The motivation for this is twofold. First, with one column only, we cover several arrival times and avoid the problem with column generations with only small

variations in arrival times. Second, with the synchronization constraints relaxed, the master problem is solvable with a wide range of well established solution methods. We use the algorithm to solve a set of test problems introduced in Bredström and Rönnqvist (2006) to optimality in 44 out of 60 cases within the time limit of one hour.

The main contributions of this work are:

- An optimal branch and price algorithm for the vehicle routing problem with synchronization constraints.
- A discussion of the advantages of including and excluding the synchronization constraints in the decomposition.
- Implicit treatment of the synchronization constraints which makes the algorithm applicable on a wide range of routing and scheduling problems, including capacitated vehicles.

The outline of this paper is as follows. In Section 2 we present a general mixed integer formulation for the synchronization problem. In Section 3 we introduce an equivalent set partitioning formulation and the relaxation of the synchronization constraints, followed by a discussion of the advantages and disadvantages of the relaxation. The branch and price algorithm is developed in Section 4. In Section 5 we present the results from runs on the test problems. We motivate the choice of branching and present an alternative branching rule. In this section we also study the integrality gap. Finally we make some concluding remarks in Section 6.

2 Problem formulation

In this paper, we concider the following variant of the vehicle routing and scheduling problem with time windows. We assume that we have a fleet of vehicles available in a depot, and a set of customers to be visited and serviced within their respective prescribed time windows. Let K denote a set of vehicles and let $G = (\bar{N}, A)$ be a directed graph, where $\bar{N} = \{o, d, 1, \ldots, n\}$ is the node set and $A = \{(i, j) \,|\, i \neq j, i \in \bar{N} \setminus \{d\}, j \in \bar{N} \setminus \{o\}\}$ is the arc set. The nodes o and d both represent the depot and the nodes $N = \{1, \ldots, n\}$ are the customers to be visited. Each customer $i \in N$ has an associated time window $[a_i, b_i]$ for the arrival time, and a duration D_i for the visit and for $i \in \{o, d\}$ the time windows $[a_i^k, b_i^k]$ define the availability for the vehicle $k \in K$. For an arc $(i, j) \in A$, we define the positive traveling time with T_{ij}.

We denote the set of pairwise synchronized visits with $P^{sync} \subset N \times N$ and call a customer i_2 virtual in a pair $(i_1, i_2) \in P^{sync}$ when i_1 and i_2 refer to one customer. In graph G, i_2 is virtual when i_2 is a duplication of the node i_1 and for each each arc to or from i_1 there is an arc to or from i_2 with equal traveling time.

The model involves two types of variables: the binary routing variables $x_{ijk} \in \{0, 1\}$ and the scheduling variables $t_{ik} \geq 0$. The routing variable x_{ijk} is one if the vehicle $k \in K$ traverses the arc $(i, j) \in A$. The scheduling variable t_{ik} is the time the vehicle k arrives at the customer $i \in N$ and is zero if the vehicle k does not visit the customer i. The formulation is as follows.

$$[P] \quad \min \sum_{k \in K} \sum_{(i,j) \in A} (C_{ik}^{Prefs} + C_{ij}^{Time}) x_{ijk} \tag{1}$$

$$s.t.$$

$$\sum_{k \in K} \sum_{j:(i,j) \in A} x_{ijk} = 1 \quad \forall i \in N \tag{2}$$

$$\sum_{j:(o,j) \in A} x_{ojk} = \sum_{j:(j,d) \in A} x_{jdk} = 1 \quad \forall k \in K \tag{3}$$

$$\sum_{j:(i,j) \in A} x_{ijk} - \sum_{j:(j,i) \in A} x_{jik} = 0 \quad i \in N \; \forall k \in K \tag{4}$$

$$t_{ik} + (T_{ij} + D_i) x_{ijk} \leq t_{jk} + b_i (1 - x_{ijk}) \quad \forall k \in K \; \forall (i, j) \in A \tag{5}$$

$$a_i \sum_{j:(i,j) \in A} x_{ijk} \leq t_{ik} \leq b_i \sum_{j:(i,j) \in A} x_{ijk} \quad \forall k \in K \; \forall i \in N \tag{6}$$

$$a_i^k \leq t_{ik} \leq b_i^k \quad \forall k \in K \; \forall i \in \{o, d\} \tag{7}$$

$$\sum_{k \in K} t_{i_1 k} = \sum_{k \in K} t_{i_2 k} \quad \forall (i_1, i_2) \in P^{sync} \tag{8}$$

$$x_{ijk} \in \{0, 1\} \quad \forall k \in K \; \forall (i, j) \in A \tag{9}$$

The objective, as seen in (1), is to minimize the sum of preferences and traveling time. The preferences C_{ik}^{Prefs} are arbitrary real values defining a weight for a specific vehicle k to service customer i. The constraints (2)-(6) form the constraint set for a multiple traveling salesman problem, where,

if we use the vocabulary of the VRSP, the constraints (2) ensure that each customer is visited by exactly one vehicle, (3) and (4) define the routing network, and the constraints (5) - (7) are the scheduling constraints. The constraint (6) implies that $t_{ik} = 0$ if customer i is not visited by the vehicle k. Therefore the arrival time for a visit i is defined by $\sum_{k \in K} t_{ik}$ which is the property we use to formulate the synchronization constraints (8). The constraints (8) ensure that the vehicles that visit the customers i_1 and i_2 for $(i_1, i_2) \in P^{sync}$ arrive simultaneously. Universally, we can model the demand of s vehicles for one customer by introducing $s-1$ virtual customers $i_2, \ldots, i_s$ and the relations (i_1, i_2), $(i_1, i_3), \ldots, (i_1, i_s) \in P^{sync}$.

3 Set partitioning formulations

We let $(A_{ij}, R_{ij})_{i \in N}$ denote a schedule $j \in J$, where $A_{ij} = 1$ when schedule j includes a service of customer i at the time R_{ij}, and $A_{ij} = R_{ij} = 0$ otherwise. If J_k is the subset of schedules feasible for vehicle k at the cost c_{jk}, the side constrained set partitioning formulation of [P] is as follows.

$$[SCSP] \quad f^*_{SCSP} = \min \sum_{k \in K} \sum_{j \in J_k} c_{kj} z_{kj} \tag{10}$$

$$s.t.$$

$$\sum_{j \in J_k} z_{kj} = 1 \quad \forall k \in K \tag{11}$$

$$\sum_{k \in K} \sum_{j \in J_k} A_{ij} z_{kj} = 1 \quad \forall i \in N \tag{12}$$

$$\sum_{k \in K} \sum_{j \in J_k} (R_{i_1 j} - R_{i_2 j}) z_{kj} = 0 \quad (i_1, i_2) \in P^{sync} \tag{13}$$

$$z_{kj} \in \{0, 1\} \quad \forall k \in K \; \forall j \in J_k \tag{14}$$

Part from the constraints in (13), $SCSP$ is a standard set partitioning model. The constraints (11) imply that one schedule is used for each vehicle and (12) that the selected schedules cover each customer exactly once. By definition of R_{ij} the arrival time at customer i equals $\sum_{k \in K} \sum_{j \in J_k} R_{ij} z_{kj}$ in a feasible solution, and we can therefore formulate the synchronization constraints with (13).

We denote with SP the relaxation of $SCSP$ obtained when the (13) are relaxed. In this paper, we aim to solve $SCSP$ with a branch and price algorithm, using the LP-relaxation SP_{LP} of SP as master problem. The feasabilty with respect to (13) is therefore treated in the branching strategies. We have two motives for approaching the relaxed problem SP. One is that with the synchronization constraints relaxed, the SP is solvable with a wide range of established solution methods. The other is that the arrival time R_{ij} is only dependent on the schedule's path and therefore we need only one column to cover several columns otherwise needed in $SCSP$.

We make the assumptions that the time windows, durations and the traveling times are integers. These assumtions imply that the variables t_{ik} in P always take integer values in a feasible solution. With these assumptions we have a valid SPP formulation of P when J is the set of all feasible schedules on the graph G satisfying the time window and traveling constraints for at least one vehicle.

3.1 The relationship between SP and $SCSP$

The formulation $SCSP$ can be viewed as an application of the integer decomposition principle (see e.g. Lübbecke and Desrosiers (2005)) on the problem P, with the columns in J_k being an enumeration of points in the bounded discrete set Γ_k of all (x_{ijk}, t_{ik}) satisfying the constraints (3)-(7), (9) and t_{ik} integer. The constraints (8) couple the constraints of the Γ_k and hence, an optimal solution to $SPSC$ may include an interior point of the convex hull $conv(\Gamma_k)$ for some k. Therefore the extreme points of $conv(\Gamma_k)$ are not enough to guarantee a valid set partitioning formulation of the problem P.

We have included the arrival times for all nodes in the definition of Γ_k to simplify the notation. This implies that for each subset $\bar{N} \subset N$, all feasible paths for the vehicle k visiting all $i \in \bar{N}$ are individually represented in Γ_k. It should be noted that it is enough to consider the arrival times for those customers included in a synchronization pair.

An example of the existance of an interior point follows. Let $K = \{1, 2\}$, $N = \{1, 2, 3\}$ and $P^{sync} = \{(1, 2)\}$. Let the vehicles have time windows such that $t_{o1} = 4$, $t_{d1} = 15$, $t_{o2} = 0$ and $t_{d2} = 10$. Further let $[a_1, b_1] = [a_2, b_2] = [6, 10]$, $[a_3, b_3] = [0, 5]$ and the durations $D_i = 0$ for all customers. The traveling time matrix is defined by $T_{o1} = T_{1d} = T_{o2} = T_{2d} = 2$, $T_{o3} = T_{3d} = 2$ and $T_{31} = T_{32} = 6$. We can assume that customer 1 and the virtual customer 2 are identical for both vehicles, and we can therefore decide for vehicle 1 to service customer 1. Therefore, vehicle 1 cannot service customer 3 and the

only feasible paths are: o – 1 – d for vehicle 1 and o – 3 – 2 – d for vehicle 2. We can deduce that the customers 1 and 2 have to be serviced at time 8 from the unique schedule for vehicle 2. Thus, there is a feasible solution where the schedule for vehicle 1 is a convex combination with weights 0.5 of the two schedules with arrival time $t_{11} = 6$ and $t_{11} = 10$ respectively, and hence not an extreme point to $conv(\Gamma_1)$.

The corresponding decomposition to obtain SP yields the sets of columns J_k from the extreme points of $conv(X_k)$, where X_k is the set of all (x_{ijk}) such that there exists (t_{ik}) with $(x_{ijk}, t_{ik}) \in \Gamma_k$. There are no feasible interior points in $conv(X_k)$. This because of the fact that with only binary variables, no interior point is integer, and all binary points are extreme. Therefore, not more than one path for each subset $\bar{N} \subset N$ for vehicle k is necessarily contained in J_k.

If we use column generation to solve the LP-relaxation $SCSP_{LP}$ of $SCSP$, we need to solve a shortest path problem with time windows (SPPTW) for each vehicle, where the dual prices from the constraints (13) appear as costs for the time variables. From the subproblem, we obtain an extreme point of $conv(\Gamma_k)$ for each k. In Ioachim et al. (1998), a dynamic programming algorithm is presented to solve the SPPTW with linear node costs in the non-elementary path case. Since the goal is to solve $SCSP$, the advantage with this approach, compared to using column generation on the LP-relaxation SP_{LP} of SP, is the possible generation of columns with multiple paths visiting the same set of customers, but with different sets of arrival times. The computational expense is a master problem with more constraints, and, if we use a node labeling algorithm to solve the subproblem, there will be generally more labels to treat.

To obtain integer feasible solutions to $SCSP$ using branch and bound, independently if we use $SCSP_{LP}$ or SP_{LP} for master problem, we need to consider a branching rule that guarantees finding interior points of $conv(\Gamma_k)$. One natural approach is to branch on the time windows. With small enough time windows, there are no alternate paths visiting one set of customers (the network is acyclic). In the case of an acyclic network, any feasible interior point $j \in conv(\Gamma_k)$ can be obtained by adjusting the arrival times by following the unique path for vehicle k that visits the customers in j. This is a central part of the algorithm presented in this paper. Note that even if we have an integer feasible solution to SP, it is not necessarily possible to adjust the arrival times for the paths in the solution to obtain a feasible solution to $SCSP$, even in the special case when the network is acyclic.

4 Branch and price algorithm

A branch and bound algorithm based on LP-relaxations solved with column generation is commonly refered to as a branch and price algorithm, a concept formalized in the paper by Barnhart et al. (1998). What foremost distinguishes a B&P algorithm from a classical B&B algorithm, is the need to use branching rules that are applicable in the context of a column generation process. For the problem in this paper, we begin by looking at the properties a solution to SP_{LP} can possess.

Solution properties

Assume that we have an optimal solution $(\bar{z}_{kj})$ to SP_{LP} with the objective function value $\bar{f}_{LP}$. Denote the index set for positive variables with $Z = \{(k,j) \,|\, \bar{z}_{kj} > 0, k \in K, j \in J_k\}$, and the index set for positive integer valued variables $I = \{(k,j) \,|\, \bar{z}_{kj} = 1, k \in K, j \in J_k\}$. Let

$$V = \{(i_1, i_2) \in P^{sync} \,|\, \exists (k_1, j_1) \in Z, \exists (k_2, j_2) \in Z : |R_{i_1 j_1} - R_{i_2 j_2}| > 0\}$$

$$W = \{i \in N \,|\, \exists (k_1, j_1) \in Z, \exists (k_2, j_2) \in Z : |R_{i j_1} - R_{i j_2}| > 0\}$$

In words, V is the set of synchronization pairs for which there are schedules used in the solution with a deviation in arrival time, and W is the set of customers for which there are schedules in the solution with a deviation in arrival time. With this notation, the solution possesses one of the following properties.

P1 $(\bar{z}_{kj})$ is feasible in $SCSP$, that is, $Z = I$ and $V = \emptyset$.

P2 $(\bar{z}_{kj})$ is feasible in SP, but not in $SCSP$, that is, $Z = I$ and $V \neq \emptyset$.

P3 $(\bar{z}_{kj})$ is fractional and $V \neq \emptyset$ and $W \neq \emptyset$.

P4 $(\bar{z}_{kj})$ is fractional and $V \neq \emptyset$ and $W = \emptyset$.

P5 $(\bar{z}_{kj})$ is fractional and $V = \emptyset$ and $W \neq \emptyset$.

P6 $(\bar{z}_{kj})$ is fractional and $V = W = \emptyset$.

Branching rules

The difficulty that arises when branching on time windows is to pick a node with a time window where columns will be ruled out from the master problem, not because of the generated R_{ij}'s, but because of the exclusion

of the column's extreme point in X_k. Gélinas et al. (1995) introduce a time window division such that a node has a candidate time window for branching only if at least one path in a fractional solution is no longer feasible in each of the obtained subproblems. If no such node is found, branching is performed on flow variables. We choose a somewhat different approach for the application in this paper. Suggested by our computational experiments, the number of generated columns after a series of constraint branches, over a vehicle / customer pair, can be significantly larger than after a series of time window branches. Therefore we choose to branch on time windows without requiring a guarantee that the first obtained subproblems have ruled out columns from the master problem solution in the current node. To achieve this, we adjust each column's arrival time to the earliest possible time, following its path, before ruling out a column in the current node and before identifying a new time window branch. For the branching in this application we define the following branching rules:

BR1 Branching on the time window for a customer i_0 such that $i_0 = \arg\max_{i \in W}\{|R_{ij_1} - R_{ij_2}| \,|\, (k_1, j_1), (k_2, j_2) \in Z\}$. If the maximum value is $m_W > 0$ and is attained with j_1 and j_2, where $R_{i_0 j_1} < R_{i_0 j_2}$, then the interval $[a_{i_0}, b_{i_0}]$ is divided in $[a_{i_0}, R_{i_0 j_1} + \lfloor m_W/2 \rfloor]$ and $[R_{i_0 j_1} + \lfloor m_W/2 \rfloor + 1, b_{i_0}]$. The rule is applicable when $W \neq \emptyset$.

BR2 Branching on time windows for synchronized customers such that $(i_1^0, i_2^0) = \arg\max_{(i_1,i_2) \in V}\{|R_{i_1 j_1} - R_{i_2 j_2}| \,|\, (k_1, j_1), (k_2, j_2) \in Z\}$. If the maximum value is $m_V > 0$ and is attained with j_1 and j_2, where $R_{i_1^0 j_1} < R_{i_2^0 j_2}$, then the interval $[a_{i_s}, b_{i_s}]$ is divided in $[a_{i_s}, R_{i_1^0 j_1} + \lfloor m_V/2 \rfloor]$ and $[R_{i_1^0 j_1} + \lfloor m_V/2 \rfloor + 1, b_{i_s}]$, for all s such that $(i_s, i_2^0) \in P^{sync}$ or $(i_1^0, i_s) \in P^{sync}$. The rule is applicable when $V \neq \emptyset$.

BR3 Branching on the vehicle / customer pair (k_0, i_0) such that $(k_0, i_0) = \arg\max_{k \in K, i \in N}\{\sum_{j \in J_k} A_{ij} \bar{z}_{kj} < 1\}$. This branching rule is applicable for the properties P3-P6, and is performed over the constraints $\sum_{j \in J_k} A_{i_0 j} z_{k_0 j} \in \{0, 1\}$.

Feasibility check

The arrival time at a customer does not affect the objective function value of $SCSP$. In a solution with property P2, we have a unique schedule for each vehicle but $V \neq \emptyset$. Following each column's path we can therefore, with x_{ijk} in P fixed accordingly, solve the resulting linear program. We refer to this LP as the Feasibility Check (FC) problem. The FC problem has

one continuous variable for each node, with lower and upper bounds from the constraints (6). It remains $|N| + |K|$ constraints from (5) and $|P^{sync}|$ constraints from (8). If FC returns a feasible solution, we have a feasible solution to $SCSP$ with the objective function value $\bar{f}_{LP}$.

Algorithm summary

1. Select a B&B-node to treat.
2. Rule out columns which are infeasible according to constraints from BR3
3. Adjust arrival times on the remaining column set
4. Solve the master problem
5. Adjust arrival times for all new columns
6. Prune if: the master problem solution is infeasible, feasible in $SCSP$ or feasible in $SCSP$ after applying FC. Go to 1.
7. For solutions with properties P2-P5, branch according to BR1 if $m_W > m_V$ and otherwise according to BR2. For P6, branch according to BR3. Go to 1.

5 Computational results

5.1 Test problems

We use the algorithm developed in this paper to solve a set of problems introduced in Bredström and Rönnqvist (2006). The test problems are generated based on an application of the homecare staff scheduling problem, where on average, ten percent of the customers need simultaneous service from two staff members. The staff members are available throughout the planning horizion of nine hours. The customer locations are uniformly distributed over a square area with the depot located in the center. The durations are randomized from a normal distribution with the goal of having a mean of five hours workload (excluding traveling time) for each staff member. The traveling time and durations are rounded to take integer values with the result of a time discretization of less than one minute. A customer i_1 with simultaneous service, is modeled with a virtual customer i_2 and $(i_1, i_2) \in P^{sync}$ and the i_1 and i_2 in all test problems are exchangable with no difference

in cost or in duration for any staff member. An overview of the problem characteristics is found in Table 1. The table shows for each instance, the

Table 1: Problem instances overview.

Instance	$\|N\|$	$\|K\|$	$\|P^{sync}\|$	$\sum D_i/\|K\|$ (h)	AvTD (h)	S (h)	M (h)	L (h)
1	20	4	2	4.9	0.22	1.5	2.1	2.9
2	20	4	2	4.2	0.20	1.7	2.2	3.0
3	20	4	2	5.3	0.21	1.5	2.4	3.0
4	20	4	2	5.9	0.29	1.8	2.9	3.9
5	20	4	2	5.0	0.21	1.3	2.1	3.2
6	50	10	5	4.7	0.25	1.4	2.3	3.1
7	50	10	5	5.0	0.23	1.6	2.5	3.4
8	50	10	5	6.2	0.23	1.5	2.4	3.2
9	80	16	8	6.1	0.21	1.5	2.3	2.9
10	80	16	8	5.1	0.17	1.6	2.6	3.6

number of customers $|N|$, the number of vehicles $|K|$, the number of synchronized customers $|P^{sync}|$, the average duration (hours) per vehicle calculated with $\sum_i D_i/|K|$ and the average traveling time (hours) to depot AvTD. The columns S, M and L the average time window size in hours. Looking at the average duration per member of staff, we note that instances 4,8 and 9 allow for much less waiting time than the other instances.

5.2 Experiment settings

The computational results were run on one Intel Xeon 2.67 GHz processor on a computer equipped with 2 GB RAM. The branch and price algorithm was implemented in the AMPL modeling environment. For all LP problems we used the CPLEX 10.0 solver. The ESPPTW-solver is a C++ implementation of the algorithm presented in Feillet et al. (2004) which was made available to AMPL as a shared library.

The master problem was solved by adding one column for all vehicles in each column generation iteration when the subproblems for each vehicle were identical. For our test problems, this is the case when we minimize traveling time and when no constraints from BR3 are applied. Otherwise, we solved one subproblem for each vehicle. The stopping criterion was for all reduced costs to be greater than -10^{-6}. The ESPPTW was only solved to optimality when no column with negative reduced costs was found using a simplified label dominance criterion, in which the labels are dominated only by the costs and time resources.

We wanted to find feasible solutions for the larger problem instances and optimal solutions if possible within the total time limit of 3600 seconds. Therefore, when selecting a node in the B&B-tree, we followed the 'best first' selection (lowest lower bound) of the next node to treat as long as no more than $|N|$-nodes were left to treat. With $|N|+1$ nodes or more, we made a depth first search on the branch added last. When a node was pruned, we picked the next node with the lowest LBD.

5.3 Results

In Tables 2 and 3 we present the results from the algorithm when the preferences and traveling time were minimized respectively. We use the following column names in the tables

Prob.	Name of problem instance from the test bed in Table 1.
LP	The optimum value of the LP-relaxation of SP.
LBD	The lower bound when time limit is reached.
UBD	The objective function value for the optimal solution or the best found solution.
nodes	The number of treated nodes in the B&B tree
FC	The number of times FC found a feasible solution to $SCSP$.
SUB tot.	Total number of subproblem solver calls.
SUB root	Number of subproblem calls in the root node.
Solve	Time (s) used by solvers (CPLEX and subproblem-solver).
Total	Total time (s), including AMPL user time. A * indicates that there are nodes left in the B&B-tree after 3600 seconds.

With the suggested algorithm, when minimizing preferences, we have solved problems 1–8 to optimality within the time limit of 3600 seconds. When our objective is to minimize traveling time, we obtain a more difficult problem. With more than four times more B&B-nodes treated, we have solved the problems 1-5, and three of the problems 6-9 to optimality. For problems 9S, 10S and 10L, we found no feasible solution in the results presented in Table 3. From Table 2, we observe that we have found feasible solutions for all problems.

If we look at the problems we have solved to optimality, 54% of the solution time is from the AMPL user time. An implementation in a non-scripting language therefore has the potential of significantly reducing the overall solution time.

In Table 3, we observe the relatively large number of B&B-nodes for the problems 1M and 1L. An improved adjustment of the arrival times is

Table 2: Solution overview for the problems when preferences are minimized.

Prob.	LP	LBD	UBD	nodes	FC	SUB tot.	SUB root	Solve	Total
1S	-118.34		-114.03	4	1	228	152	0.73	1.27
1M	-124.17		-117.80	7	2	263	155	0.98	1.68
1L	-124.17		-118.51	8	1	306	178	1.50	2.55
2S	-92.09		-92.09	1	0	153	153	0.34	0.60
2M	-107.15		-104.81	8	2	282	170	1.36	2.30
2L	-109.24		-107.64	21	2	508	164	3.84	6.44
3S	-100.69		-99.49	13	1	242	124	1.00	1.66
3M	-110.06		-106.59	5	1	280	161	1.19	2.01
3L	-113.78		-107.87	9	1	329	162	1.64	2.63
4S	-106.24		-100.00	8	2	261	124	1.00	1.72
4M	-110.28		-106.72	9	2	300	152	1.42	2.36
4L	-111.74		-109.27	18	1	459	158	2.95	5.04
5S	-76.29		-76.29	1	1	139	139	0.34	0.64
5M	-77.03		-76.29	3	2	225	180	0.72	1.28
5L	-84.66		-84.21	6	1	275	164	1.30	2.21
6S	-372.21		-370.06	51	1	1578	864	78.45	150.63
6M	-386.86		-379.88	101	2	1861	875	126.48	247.88
6L	-398.70		-387.20	154	5	2495	818	244.94	474.15
7S	-404.85		-401.11	103	4	2066	775	146.26	291.29
7M	-406.17		-406.17	22	1	1240	721	49.53	86.70
7L	-414.76		-407.48	264	4	3030	731	368.32	714.62
8S	-383.66		-380.76	112	1	1223	718	68.34	135.39
8M	-405.87		-403.57	166	2	1851	796	148.11	290.77
8L	-410.43		-407.48	206	2	2032	731	197.17	362.18
9S	-639.26	-633.91	-552.65	414	0	5773	1922	1892.92	*
9M	-664.08	-661.77	-463.82	412	0	6026	1808	1955.00	*
9L	-675.61	-673.09	-663.47	368	3	5164	1901	1856.07	*
10S	-680.53	-676.14	-675.81	796	1	3744	2167	1732.41	*
10M	-694.97	-687.90	-685.31	328	9	5341	2091	1807.09	*
10L	-700.03	-693.98	-691.34	433	3	4764	1973	1807.72	*

suggested in Section 5.5 which reduces the number of treated nodes.

5.4 Branching strategy

In Figures 1 and 2, we illustrate typical examples of the solution process when, in case of a fractional solution, we choose to branch with BR3 first and when we use the branching order the algorithm suggested in Section 4, that is BR3 last. In both figures, the x-axis shows the number of B&B-nodes and the y-axis the number of subproblem solver calls. In the case of minimizing traveling time, when branching with BR3 first, the algorithm finds no solution and terminates with an LBD of 8.145 after 8998 subproblem calls, and a total of 152 B&B-nodes. With BR3 last, the algorithm terminates with an LBD of 8.527 and a feasible solution with a value of 8.540. It had

Table 3: Solution overview for the problems traveling time is minimized.

Prob.	LP	LBD	UBD	nodes	FC	SUB tot.	SUB root	Solve	Total
1S	3.548		3.548	17	1	159	83	1.01	1.96
1M	3.475		3.548	2192	0	595	83	103.85	221.93
1L	3.081		3.393	1059	1	568	96	50.27	107.41
2S	4.266		4.266	17	0	299	99	2.02	3.28
2M	3.546		3.575	56	1	417	96	4.64	8.12
2L	3.199		3.415	9	1	210	100	1.50	2.72
3S	3.526		3.628	87	2	533	75	7.80	14.17
3M	3.291		3.333	151	4	537	85	9.49	17.57
3L	3.227		3.295	420	4	768	89	22.46	42.78
4S	6.076		6.141	152	1	356	61	7.04	14.02
4M	5.387		5.670	214	4	530	74	13.88	27.53
4L	4.711		5.134	66	1	380	85	5.04	9.74
5S	3.724		3.929	18	1	209	80	1.56	2.84
5M	3.490		3.527	550	0	506	89	26.86	57.04
5L	3.176		3.339	66	2	352	88	4.78	9.11
6S	8.083	8.129	8.143	2887	2	1244	280	2500.79	*
6M	7.375	7.674	7.714	1250	0	1955	304	2468.43	*
6L	6.807		7.138	916	3	2242	341	1874.08	3279.48
7S	8.223		8.392	525	0	1782	287	805.25	1472.39
7M	7.112	7.355	7.673	1089	1	2229	278	2376.01	*
7L	6.537	6.871	6.885	804	2	2720	304	2104.61	*
8S	9.169		9.537	260	4	1656	266	487.74	931.95
8M	8.145	8.527	8.540	917	1	2342	263	2231.23	*
8L	7.624	7.913	8.616	894	0	2168	266	2377.05	*
9S	11.498	11.695	-	437	0	2252	281	1991.10	*
9M	10.635	10.792	11.745	403	0	2378	477	1931.61	*
9L	10.159	10.366	11.106	362	0	2686	471	1938.34	*
10S	8.204	8.398	-	304	0	2548	560	1840.07	*
10M	7.392	7.502	8.538	630	0	2626	565	1606.39	*
10L	6.876	7.054	-	215	0	3025	600	1592.40	*

made 2342 subproblem calls in a total of 917 treated B&B nodes.

In the case of minimizing preferences, both branching strategies found the optimal solution. When branching with BR3 first, the optimalty was proven after 2064 seconds in 158 B&B-nodes with 5278 generated columns. When branching on BR3 last, the optimalty was proven after 291 seconds in 166 B&B-nodes and 1851 calls to the subproblem solver.

5.5 Enhanced arrival time adjustment method

For some problems, we obtained a B&B-tree with a significantly above average number of nodes, without having a proportionally larger number of subproblem calls. We therefore modified the algorithm to find a more significant time window to branch on in each iteration. We replaced Step 5

Figure 1: Subproblem calls in each node for the problem 8M when minimizing traveling time.

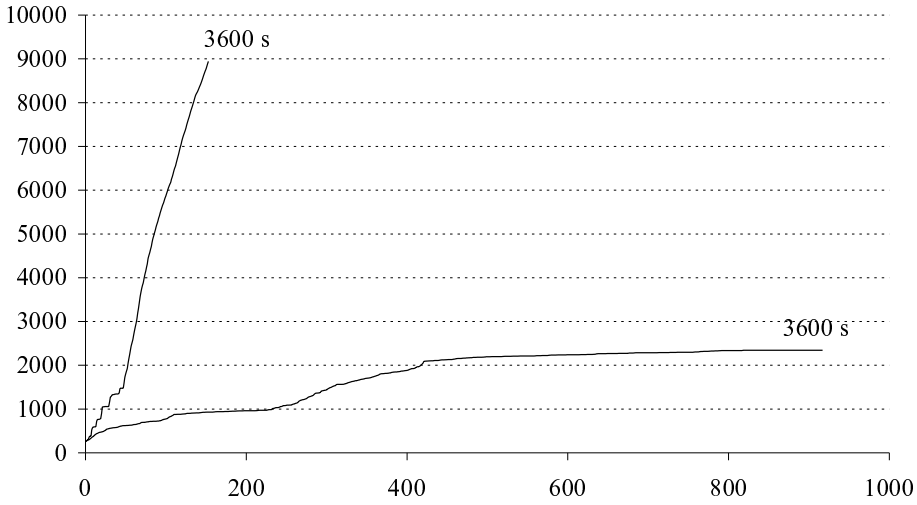
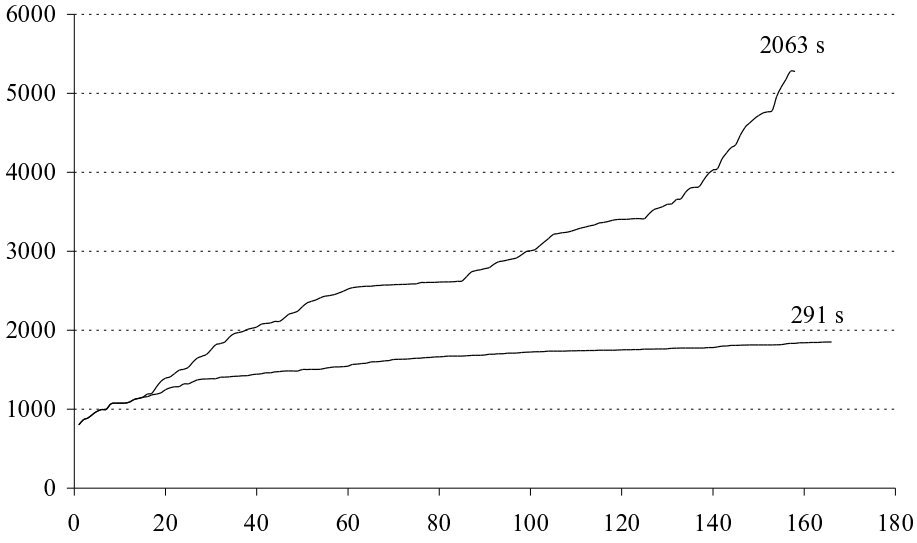


Figure 2: Subproblem calls in each node for the problem 8M when minimizing preferences.



in the algorithm with an LP problem, with the objective of minimizing the deviation of arrival time for each node. For this problem, we defined a time variable δ_i for each node, and one time variable for each node serviced by a column j, for each column j such that there is a $(k, j) \in Z$. The constraints were that the columns were individually feasible given the current time windows. With variables and constraints measuring the deviation from δ_i, we minimize the sum of deviations, and the solution obtained is the adjusted arrival times for the columns represented in Z.

The results of this modification used when we minimize traveling time is presented in Table 4. With the exeption of the problems 2L and 5S, both

Table 4: Solution overview from the problems with service time adjusted with LP before branching.

Prob.	LP	LBD	UBD	nodes	FC	SUB tot.	SUB root	Solve	Total
1S	3.548		3.548	5	1	165	83	0.65	1.12
1M	3.475		3.548	31	2	223	83	1.59	3.69
1L	3.081		3.393	128	1	335	96	3.74	11.91
2S	4.266		4.266	1	1	99	99	0.28	0.56
2M	3.546		3.575	30	1	240	96	1.42	3.2
2L	3.199		3.415	32	2	470	100	3.58	7.41
3S	3.526		3.628	21	1	269	75	1.71	3.84
3M	3.291		3.333	18	1	350	85	2.17	4.31
3L	3.227		3.295	5	1	146	89	0.74	1.44
4S	6.076		6.141	10	1	149	61	0.67	1.54
4M	5.387		5.670	11	1	209	74	1.17	2.55
4L	4.711		5.134	38	1	363	85	3.19	7.69
5S	3.724		3.929	13	2	213	80	1.32	2.9
5M	3.490		3.527	152	0	254	89	2.68	9.1
5L	3.176		3.339	36	1	270	88	2.07	5.15
6S	8.083		8.143	58	1	1782	280	102.50	197.92
6M	7.375		7.701	2046	1	1728	304	1738.52	*
6L	6.807	7.125	7.138	994	1	2966	341	1680.51	*
7S	8.223		8.392	89	1	1304	287	85.52	169.3
7M	7.112	7.410	7.562	1413	1	2036	278	1908.59	*
7L	6.537	6.857	6.885	768	2	3534	304	1522.65	*
8S	9.169		9.537	286	2	1613	266	421.72	850.52
8M	8.145		8.540	946	2	2886	263	1780.48	3490.57
8L	7.624	7.939	8.105	1330	1	2491	266	1922.87	*
9S	11.498	11.682	12.209	358	0	3265	281	2172.11	*
9M	10.635	10.792	11.043	333	0	3323	477	2116.79	*
9L	10.159	10.366	10.893	489	0	2195	471	1948.49	*
10S	8.204	8.426	9.126	402	0	3653	560	1941.86	*
10M	7.392	7.516	8.104	308	0	3731	565	1761.80	*
10L	6.876	7.054	-	288	0	4300	600	1630.70	*

the number of B&B-nodes and the total time was reduced for problems 1–5.

Overall for these problems, the total solution time was reduced from 540 to 66 seconds. With the modification, we also found two new optimal solutions among problems 6–8 and feasible solutions to 5 out of the 6 problems in 9–10.

5.6 Characteristics

We want to see the effect of the synchronization constraints in the problems in our test bed. To do this, we compare the gap from the optimal value of SP_{LP} with the optimal integer solutions with and without the synchronization constraints relaxed. The results from this comparison are shown in Table 5. The columns VRP are the gap with relaxed synchronization,

Table 5: Integrality gap calculated as $100(f^* - f_{LP})/f_{LP}$.

Problem	Preferences		Time	
	VRP	Sync	VRP	Sync
1S	0.00	3.64	0.00	0.00
1M	0.00	5.13	2.11	2.11
1L	0.00	4.56	8.86	10.12
2S	0.00	0.00	0.00	0.00
2M	0.00	2.18	0.14	0.83
2L	0.00	1.46	0.19	6.74
3S	0.00	1.20	0.00	2.91
3M	0.00	3.15	1.29	1.29
3L	0.00	5.19	2.10	2.10
4S	0.00	5.87	0.80	1.07
4M	0.00	3.23	0.42	5.25
4L	0.00	2.21	1.72	8.97
5S	0.00	0.00	1.12	5.51
5M	0.00	0.97	1.08	1.08
5L	0.00	0.53	0.00	5.14
6S	0.00	0.58	0.58	0.75
6M	0.09	1.80	0.79	4.42
6L	0.00	2.88	0.54	4.87
7S	0.00	0.92	0.54	2.05
7M	0.00	0.00	1.08	*
7L	0.00	1.75	0.67	*
8S	0.00	0.76	0.80	4.01
8M	0.00	0.57	0.76	4.84
8L	0.00	0.72	*	*

and the columns Sync the gap without relaxed constraints. We observe that even when minimizing preferences, we obtain a significant integrality gap

even though in most cases the underlying VRPs have no gap at all. In 9 out of 45 problems the gap is unchanged, but for the 24 preference problems the average gap increase is 205% and the 21 time problems have an average gap increase of 239%.

For the test problems in this paper we have the additional property that the integrality gap equals the integrality gap for $SCSP$. This is because the synchronization is for customers requiring identical multiple visits only. That is, for all $(i_1, i_2) \in P^{sync}$, i_2 is a virtual customer with $D_{i_2} = D_{i_1}$ and $C^{Prefs}_{i_2 k} = C^{Prefs}_{i_1 k}$ for all $k \in K$. In the subproblem solver, we exclude routes with visits to both i_1 and i_2, for all $(i_1, i_2) \in P^{sync}$. This is equal to including the constraints $\sum_{(i_1,j)\in A} x_{i_1 jk} + \sum_{(i_2,j)\in A} x_{i_2 jk} \leq 1$ for all $(i_1, i_2) \in P^{sync}$ in the definitions of Γ_k and X_k. These constraints are redundant in the formulation of P since they follow directly from (8) and the assumption that all $T_{ij} > 0$. Therefore, for any extreme point j to $conv(X_k)$, there exists an extreme point j' to $conv(X_k)$, where all customers i_1 in j, that have a virtual customer i_2, are replaced with its virtual customer, and vice versa. The corresponding columns to j and j' in SP have equal costs for each k, that is $c_{jk} = c_{j'k}$, and from any feasible solution (z_{kj}) to SP_{LP} we can construct a feasible solution $(\bar{z}_{kj})$ to SP_{LP} by replacing each j for j'. It follows that the convex combination with weights 0.5 of the solutions (z_{kj}) and $(\bar{z}_{kj})$ is feasible in SP_{LP}, and also in $SCSP_{LP}$.

Since all feasible solutions to SP_{LP} are feasible in $SCSP_{LP}$, it follows that the gap between $SCSP$ and SP_{LP} is not greater than the integrality gap for $SCSP$, since SP_{LP} is a relaxation of $SCSP_{LP}$.

6 Concluding remarks

We consider the combined vehicle routing and scheduling problem with synchronization constraints, which is motivated from the range of practical applications which have vehicle synchronization constraints. The results show that synchronization can make a problem significantly harder, in the sense of an increased integrality gap.

The algorithm's performance is highly dependent on the branching strategy. In the results, we find that for our test problems, time window branching is preferable to constraint branching as long as time window branches can be found.

To further improve the algorithm, we introduce an enhanced method to adjust the arrival times. The method successfully improved the solution times and reduced the number of branch and bound nodes. The algorithm

introduced in this paper solved 44 out of the 60 test problems where two were only found using the enhanced method.

It is worth noting that the synchronization constraints can be formulated with two temporal precedence constraints with the offsets $S_{i_1 i_2} = S_{i_2 i_1} = 0$ defined by

$$\sum_{k \in K} t_{i_1 k} \leq S_{i_1 i_2} + \sum_{k \in K} t_{i_2 k} \quad \forall (i_1, i_2) \in P^{prec}. \tag{15}$$

The constraint says that for each pair $(i_1, i_2) \in P^{prec}$ the temporal offset $S_{i_1 i_2}$ sets the requirement that i_2 is visited at least $S_{i_1 i_2}$ time units after i_1. With minor modifications, the algorithm introduced in Section 4 can be applied on the more general problem where (8) is replaced by (15).

References

C. Barnhart, E. Johnson, G. Nemhauser, and M. W. P. Savelsbergh. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–332, 1998.

J. Beck, P. Prosser, and E. Selensky. Vehicle routing and job shop scheduling: What's the difference? *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS03)*, 2003.

D. Bredström and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. Technical report, Scandinavian Working Papers in Economics, NHH Discussion Paper 18/2006, 2006.

J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. *Network Routing*, volume 8 of *Handbook in Operations Research and Management Science*, chapter Time Constrained Routing and Scheduling, pages 35–139. North-Holland, 1995.

P. Eveborn, P. Flisberg, and M. Rönnqvist. Laps care - and operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962–976, 2006.

K. Fagerholt and M. Christiansen. A travelling salesman problem with allocation, time window and precedence constraints — an application to ship scheduling. *International Transactions in Operational Research*, 7 (3):231–244, 2000.

D. Feillet, P. Dejax, M. Gendreau, and C. Guenguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

R. Freling, D. Huisman, and A. P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6: 63–85, 2003.

S. Gélinas, M. Desrochers, J. Desrosiers, and M. M. Solomon. A new branching strategy for time constrained routing problems with applications to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

K. Haase, G. Desaulniers, and J. Desrosiers. Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, 35(3): 286–303, 2001.

I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31:193–204, 1998.

I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119:75–90, 1999.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38(2):197–209, 2004.